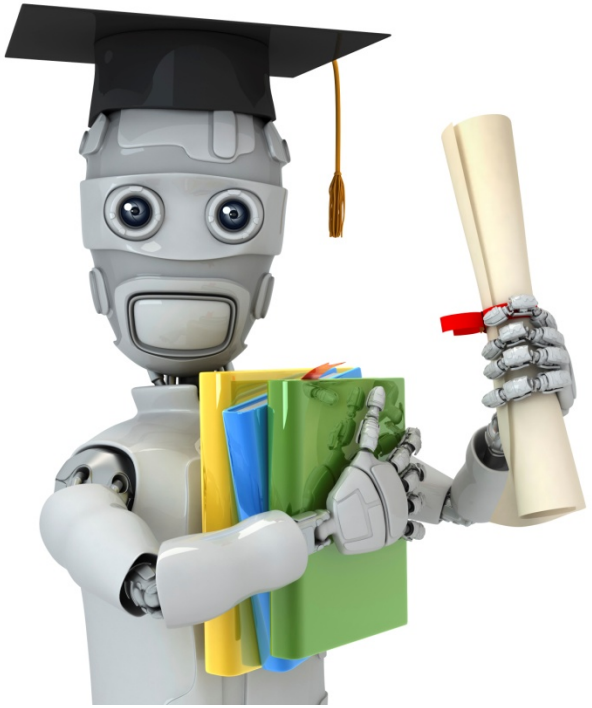


Machine Learning

# Logistic Regression

## Classification

# Classification

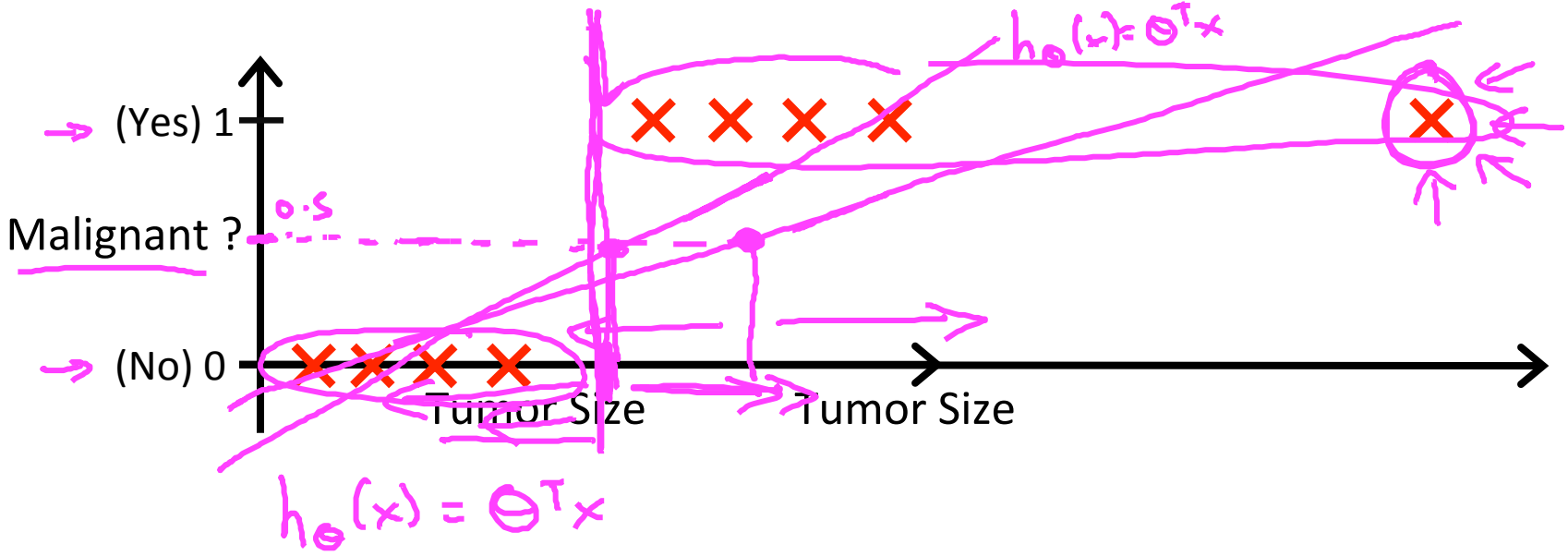
- Email: Spam / Not Spam?
- Online Transactions: Fraudulent (Yes / No)?
- Tumor: Malignant / Benign ?

$y \in \{0, 1\}$

0: “Negative Class” (e.g., benign tumor)

1: “Positive Class” (e.g., malignant tumor)

$y \in \{0, 1, 2, 3\}$



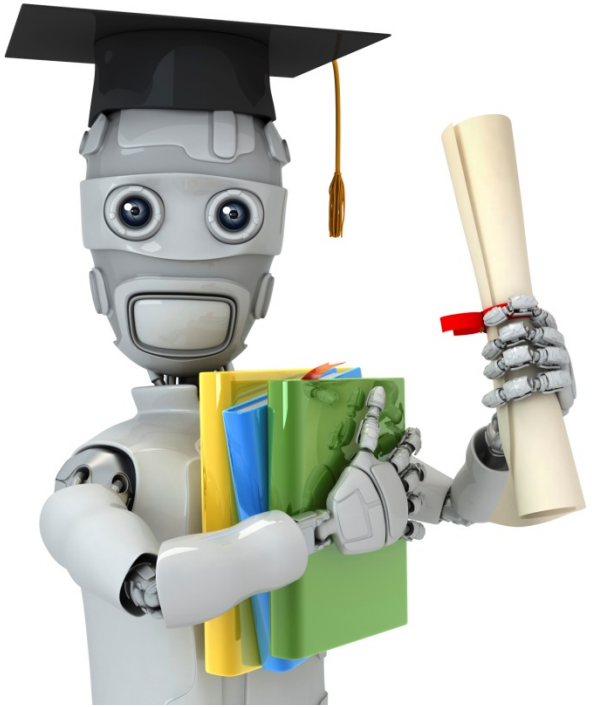

- Threshold classifier output $h_\theta(x)$ at 0.5:
  - If $h_\theta(x) \geq 0.5$, predict "y = 1"
  - If $h_\theta(x) < 0.5$, predict "y = 0"

Classification: $y = 0$ or $1$

$h_\theta(x)$ can be > 1 or < 0

Logistic Regression: $0 \leq h_\theta(x) \leq 1$

Classification



Machine Learning

# Logistic Regression

## Hypothesis Representation

## Logistic Regression Model

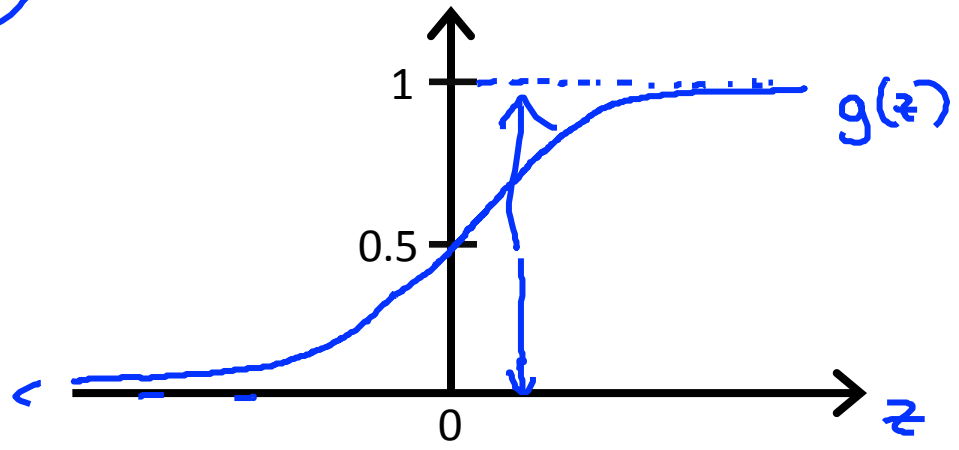
Want $0 \leq h_\theta(x) \leq 1$

$$h_\theta(x) = g(\theta^T x)$$

$$g(z) = \frac{1}{1+e^{-z}}$$

$z = \theta^T x$

$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$




- Sigmoid function
- Logistic function

Parameters $\theta$.

## Interpretation of Hypothesis Output

$h_\theta(x)$

$h_\theta(x)$ = estimated probability that y = 1 on input x

Example: If $x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{tumorSize} \end{bmatrix}$

$h_\theta(x) = 0.7$ $\quad$ $y=1$

Tell patient that 70% chance of tumor being malignant

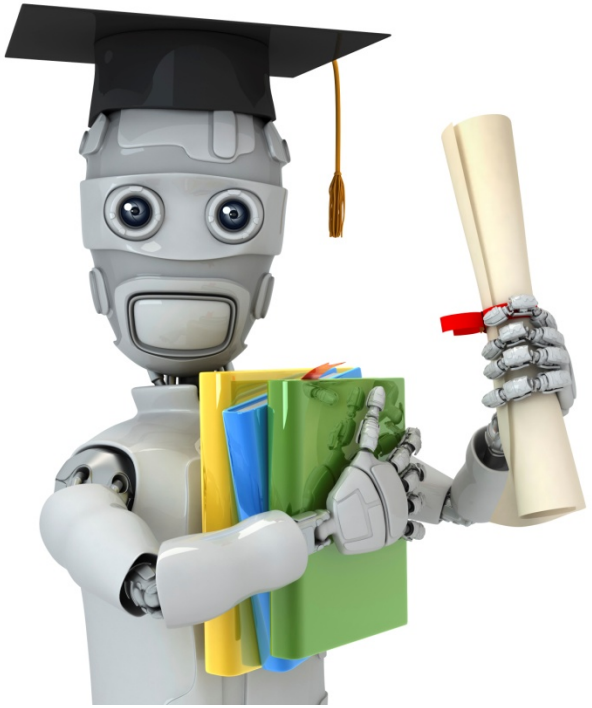
$h_\theta(x) = P(y=1|x;\theta)$

$y = 0$ or $1$

"probability that y = 1, given x, parameterized by $\theta$"

$P(y=0|x;\theta) + P(y=1|x;\theta) = 1$

$P(y=0|x;\theta) = 1 - P(y=1|x;\theta)$



Machine Learning

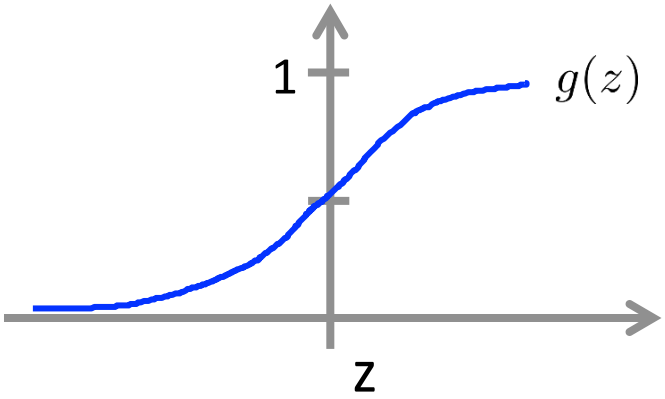
# Logistic Regression

## Decision boundary

## Logistic regression

$$h_\theta(x) = g(\theta^T x)$$

$$g(z) = \frac{1}{1+e^{-z}}$$




Suppose predict "$y = 1$" if $h_\theta(x) \geq 0.5$

$\theta^T x \geq 0$

$g(z) \geq 0.5$ when $z \geq 0$

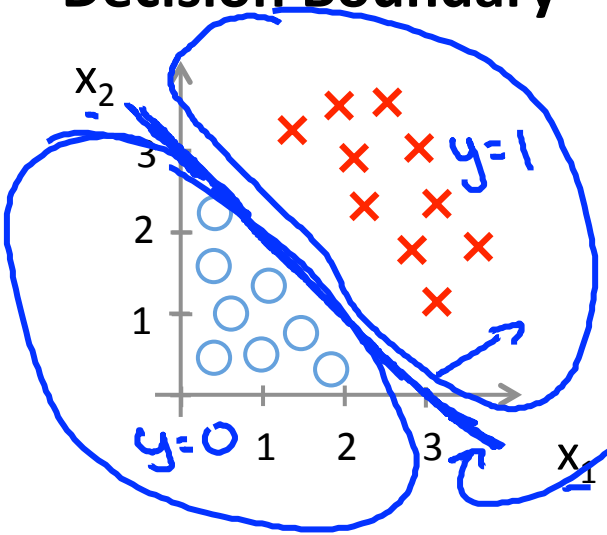
$h_\theta(x) = g(\theta^T x)$

predict "$y = 0$" if $h_\theta(x) < 0.5$

$\theta^T x < 0$

$g(z) < 0.5$ when $z < 0$

# Decision Boundary




$\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$

$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

$-3$ $1$ $1$

Decision boundary

Predict "$y = 1$" if $-3 + x_1 + x_2 \geq 0$

$\theta^T x$

$x_1 + x_2 \geq 3$

$x_1, x_2$

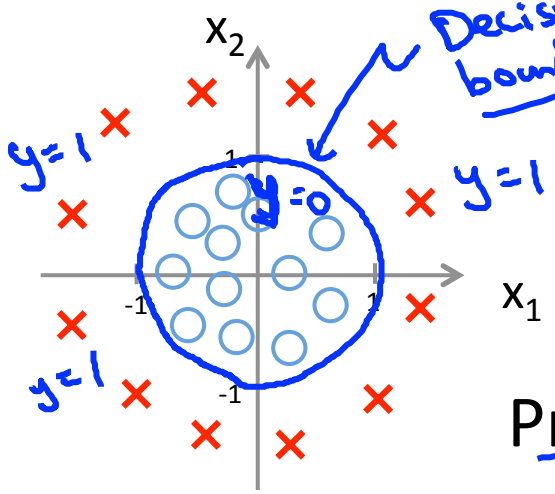
$h_\theta(x) = 0.5$

$x_1 + x_2 = 3$

$x_1 + x_2 < 3$

$y = 0$

# Non-linear decision boundaries



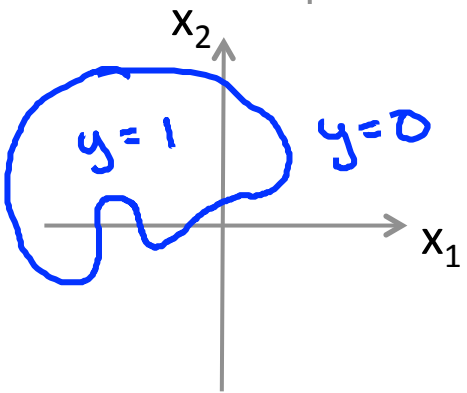

$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

$$\theta = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

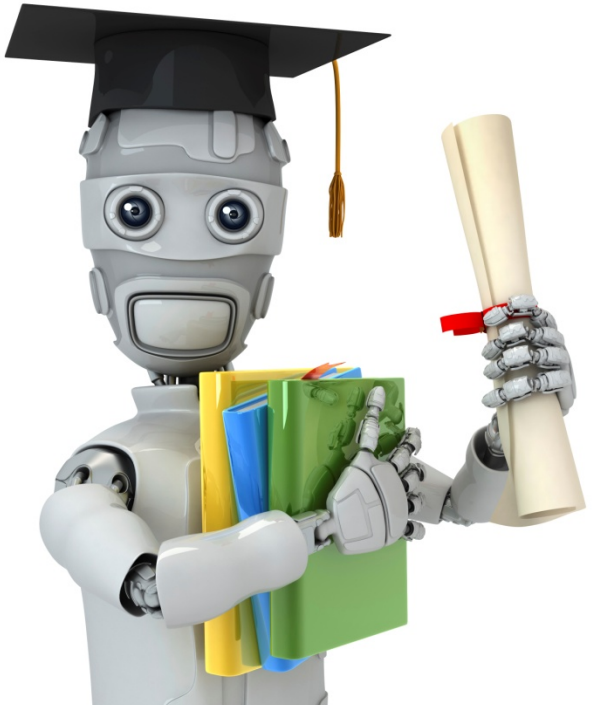
Predict "$y = 1$" if $-1 + x_1^2 + x_2^2 \geq 0$

$x_1^2 + x_2^2 = 1$

$x_1^2 + x_2^2 \geq 1$




$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1^2 x_2 + \theta_5 x_1^2 x_2^2 + \theta_6 x_1^3 x_2 + \dots)$$



Machine Learning

# Logistic Regression

## Cost function

Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})\}$

m examples $x \in \begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_n \end{bmatrix}$ $\mathbb{R}^{n+1}$ $x_0 = 1, y \in \{0, 1\}$

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

How to choose parameters $\theta$ ?

# Cost function

Linear regression:

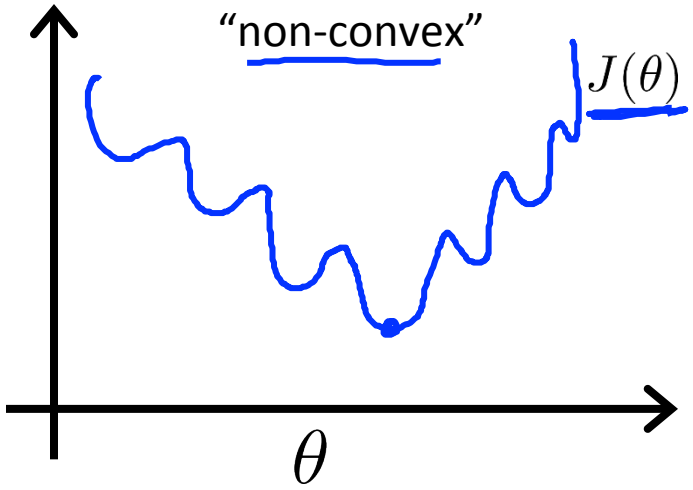
$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2} \left( h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

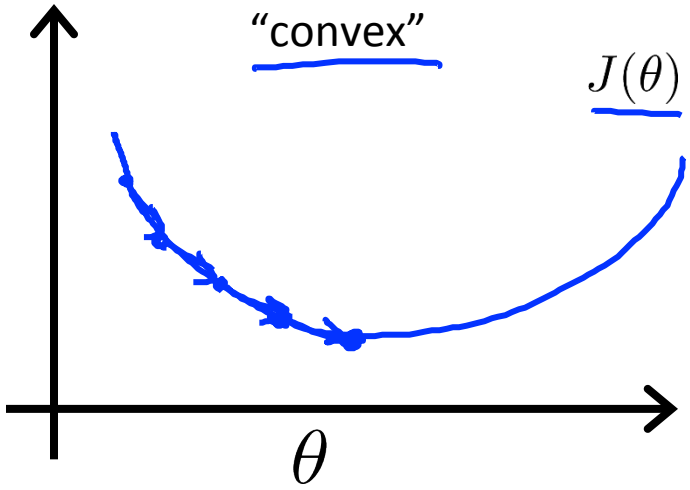
Logistic

$$\text{Cost}(h_\theta(x^{(i)}), y^{(i)}) = \frac{1}{2} \left( h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

$\text{cost}(h_\theta(x^{(i)}), y)$

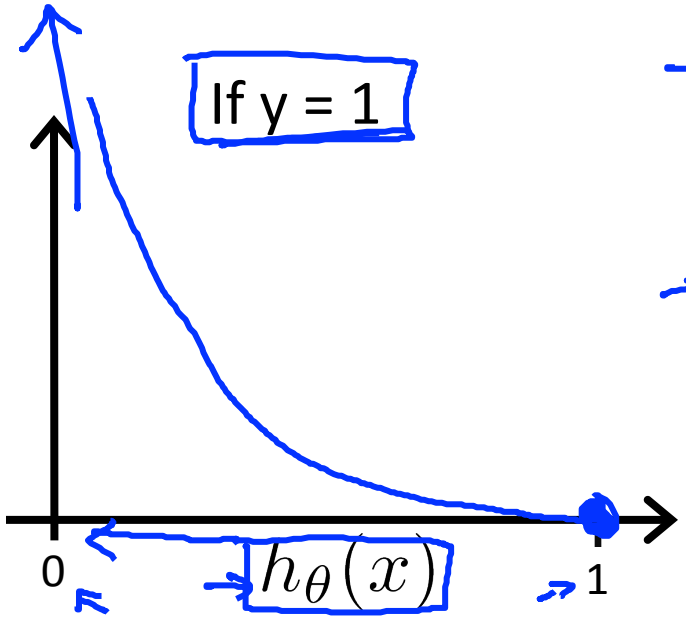
$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$



# Logistic regression cost function

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

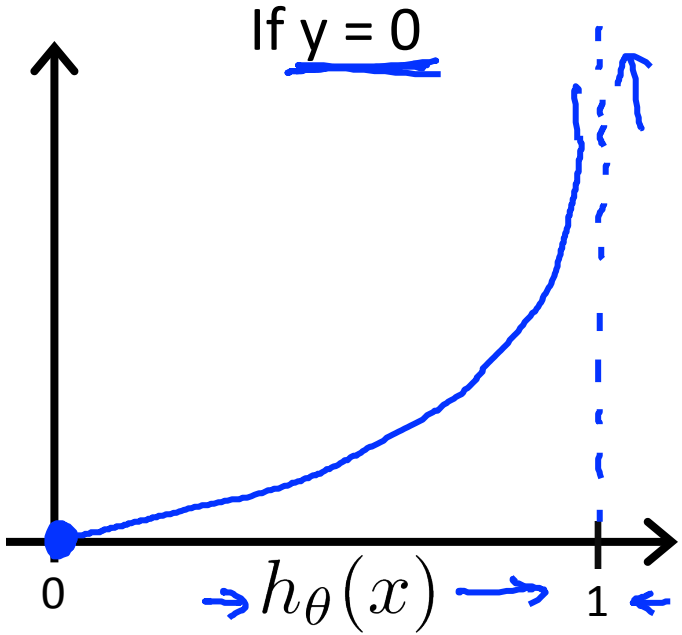


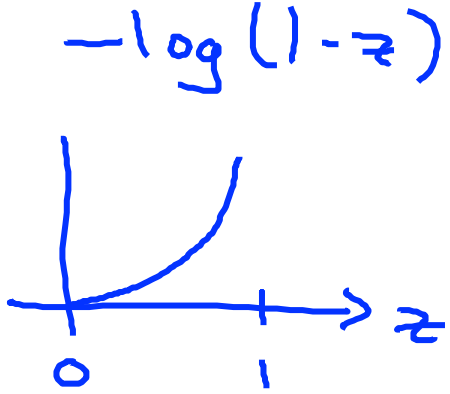

- $\text{Cost} = 0$ if $y = 1, h_\theta(x) = 1$
  But as $h_\theta(x) \to 0$
  $Cost \to \infty$

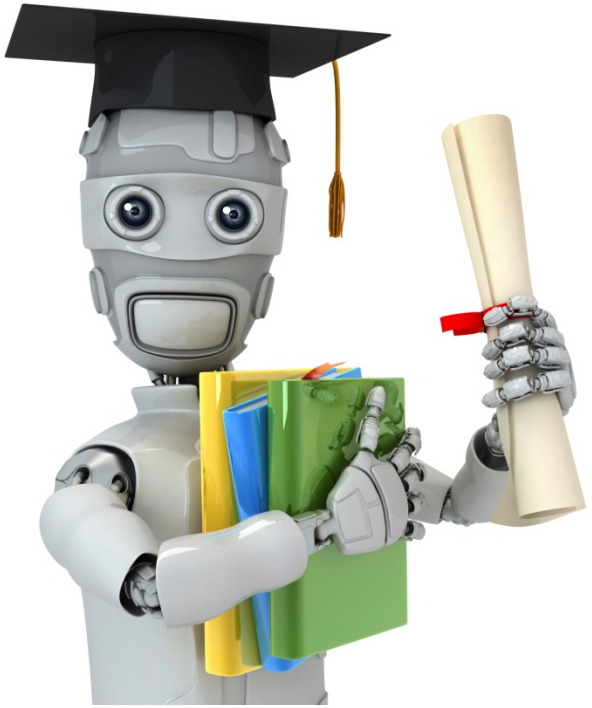
- Captures intuition that if $h_\theta(x) = 0$, (predict $P(y = 1|x; \theta) = 0$), but $y = 1$, we'll penalize learning algorithm by a very large cost.

# Logistic regression cost function

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$





Machine Learning

# Logistic Regression

## Simplified cost function and gradient descent

# Logistic regression cost function

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

Note: $y = 0$ or $1$ always

$$\text{Cost}(h_\theta(x), y) = -y \log(h_\theta(x)) - (1-y) \log(1 - h_\theta(x))$$

If $y = 1$: $\text{Cost}(h_\theta(x), y) = -\log h_\theta(x)$

If $y = 0$: $\text{Cost}(h_\theta(x), y) = -\log(1 - h_\theta(x))$

## Logistic regression cost function

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

$$= -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

To fit parameters $\theta$ :

$$\min_\theta J(\theta)$$

Get $\theta$

To make a prediction given new $x$:

Output $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$

$p(y=1 \mid x;\theta)$

## Gradient Descent

$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

Want $\min_\theta J(\theta)$:

Repeat $\{$

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

(simultaneously update all $\theta_j$)

$\}$

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

# Gradient Descent

$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

Want $\min_\theta J(\theta)$:

Repeat $\{$

$$\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

(simultaneously update all $\theta_j$)

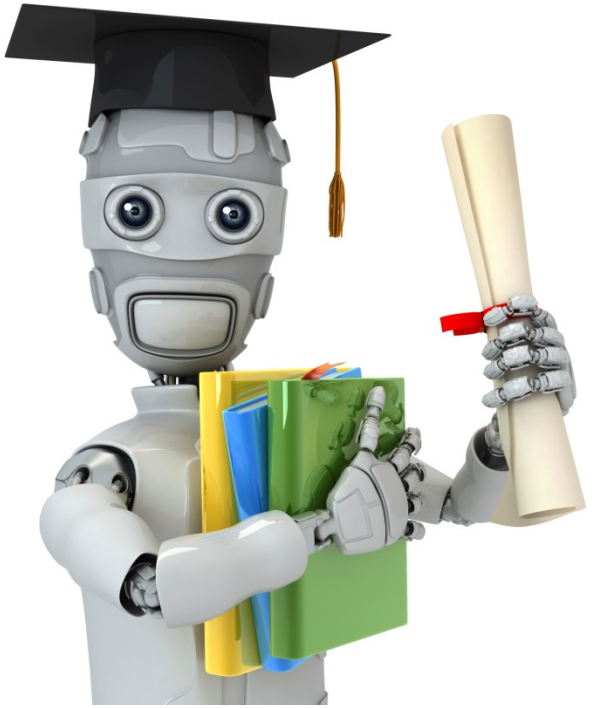
$\}$

$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}$ for $i = 0$ to $n$

$h_\theta(x) = \theta^T x$

$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$

Algorithm looks identical to linear regression!



Machine Learning

# Logistic Regression

# Advanced optimization

## Optimization algorithm

Cost function $J(\theta)$. Want $\min_\theta J(\theta)$.

Given $\theta$, we have code that can compute

- $J(\theta)$
- $\frac{\partial}{\partial \theta_j} J(\theta)$ (for $j = 0, 1, \ldots, n$ )

Gradient descent:

Repeat $\{$

$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$

$\}$

## Optimization algorithm

Given $\theta$, we have code that can compute

- $J(\theta)$
- $\frac{\partial}{\partial \theta_j} J(\theta)$

(for $j = 0, 1, \ldots, n$ )

Optimization algorithms:

- Gradient descent
- Conjugate gradient
- BFGS
- L-BFGS

Advantages:

- No need to manually pick $\alpha$
- Often faster than gradient descent.

Disadvantages:

- More complex

Example: $\min_\theta J(\theta)$

$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$ $\theta_1 = 5, \theta_2 = 5.$

$J(\theta) = (\theta_1 - 5)^2 + (\theta_2 - 5)^2$

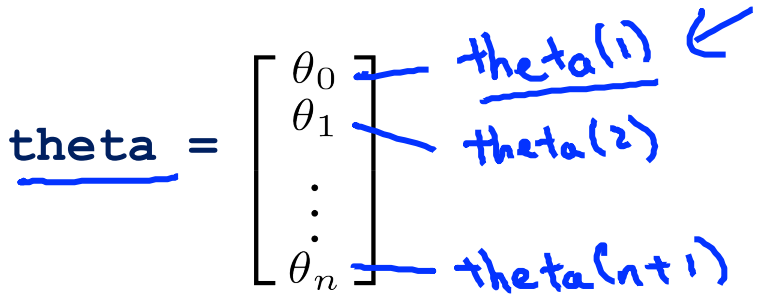
$\frac{\partial}{\partial \theta_1} J(\theta) = 2(\theta_1 - 5)$

$\frac{\partial}{\partial \theta_2} J(\theta) = 2(\theta_2 - 5)$

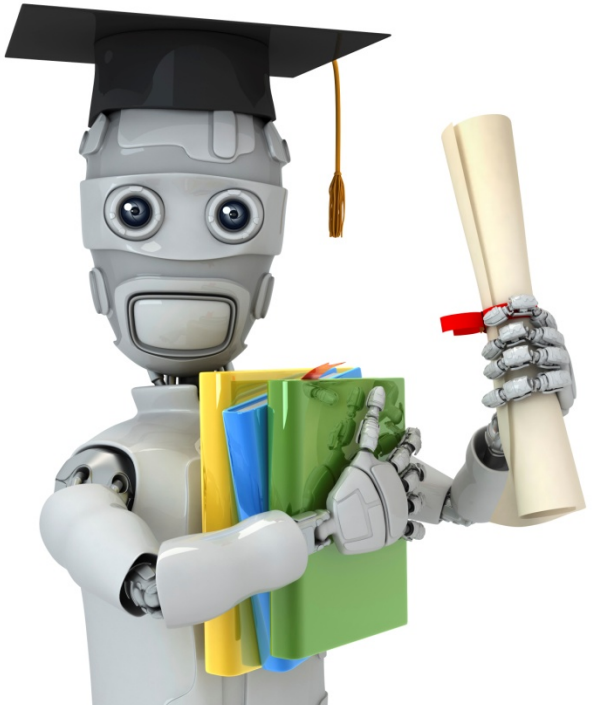
```
function [jVal, gradient]
            = costFunction(theta)
  jVal = (theta(1)-5)^2 + ...
         (theta(2)-5)^2;
  gradient = zeros(2,1);
  gradient(1) = 2*(theta(1)-5);
  gradient(2) = 2*(theta(2)-5);
```

```
options = optimset('GradObj', 'on', 'MaxIter', '100');
initialTheta = zeros(2,1);
[optTheta, functionVal, exitFlag] ...
    = fminunc(@costFunction, initialTheta, options);
```

$\theta \in \mathbb{R}^d \quad d \geq 2$.




```
function [jVal, gradient] = costFunction(theta)
  jVal = [code to compute J(theta)];
  gradient(1) = [code to compute ∂/∂θ0 J(θ)];
  gradient(2) = [code to compute ∂/∂θ1 J(θ)];
  ...
  gradient(n+1) = [code to compute ∂/∂θn J(θ)];
```



Machine Learning

# Logistic Regression

## Multi-class classification: One-vs-all

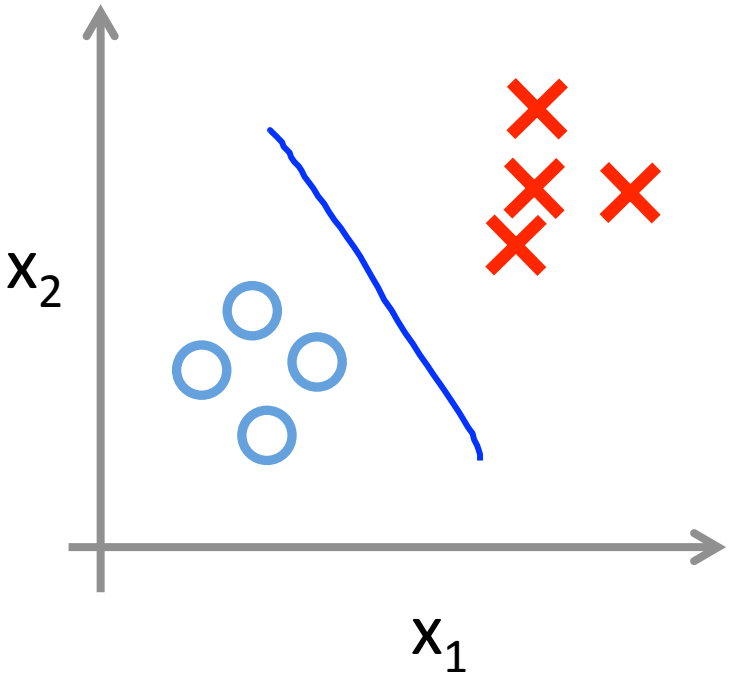
# Multiclass classification

Email foldering/tagging: Work, Friends, Family, Hobby

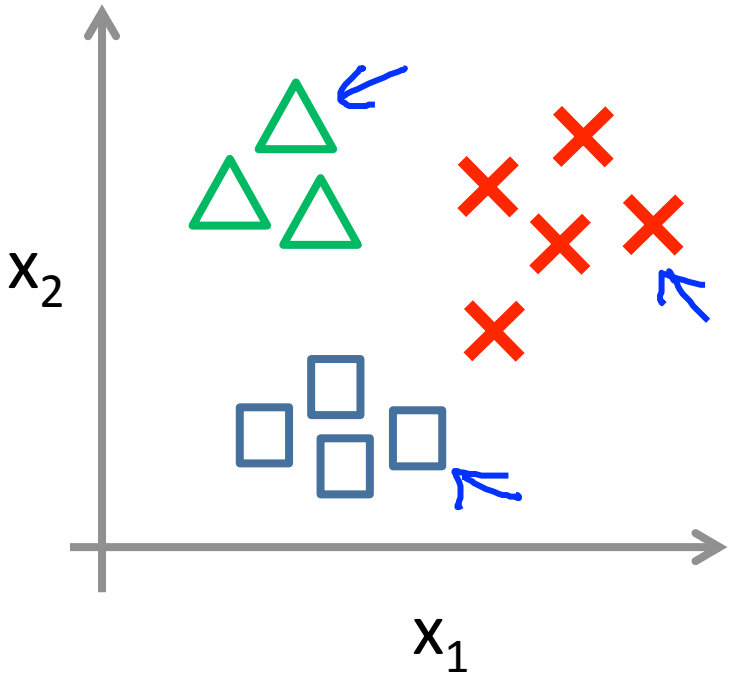
Medical diagrams: Not ill, Cold, Flu

Weather: Sunny, Cloudy, Rain, Snow

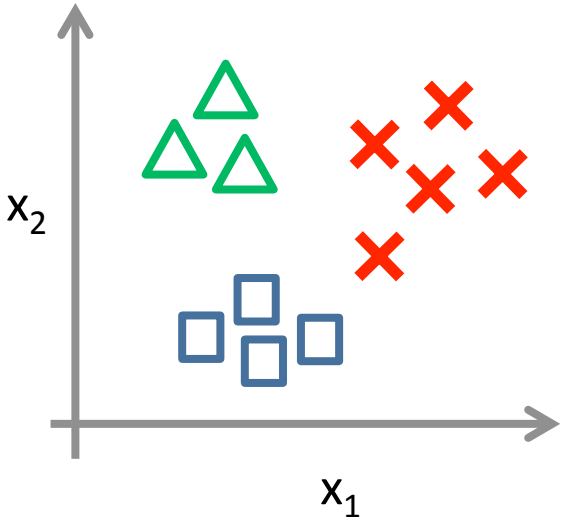
Binary classification:




Multi-class classification:



# One-vs-all (one-vs-rest):

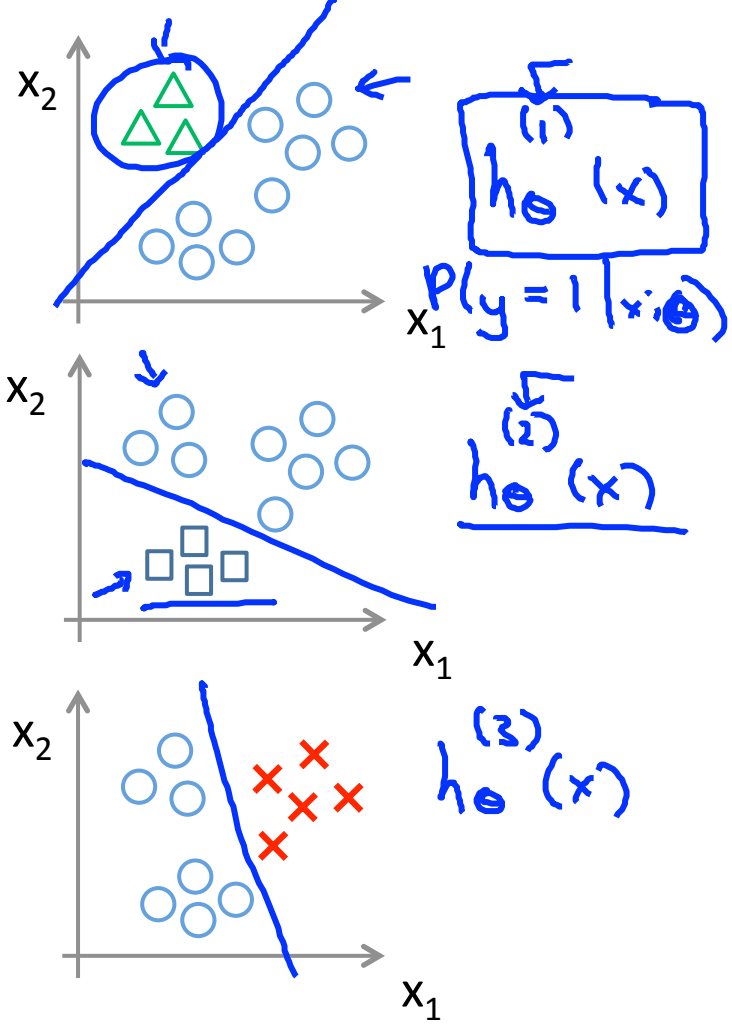



Class 1: △

Class 2: □

Class 3: ×

$$h_\theta^{(i)}(x) = P(y = i|x;\theta) \qquad (i = 1, 2, 3)$$



# One-vs-all

Train a logistic regression classifier $h_\theta^{(i)}(x)$ for each class $i$ to predict the probability that $y = i$.

On a new input $x$, to make a prediction, pick the class $i$ that maximizes

$$\max_i h_\theta^{(i)}(x)$$